Software Requirements Specification (SRS)

# Project Title: WebArtifacts IT Solutions

1. Introduction  
1.1 Purpose  
This SRS document defines the complete functional and non-functional requirements for the dynamic website of WebArtifacts, a startup IT services company. The site will serve both as a public-facing platform for clients and a secure portal for employees. The document is intended for stakeholders, developers, testers, and future maintainers.  
  
1.2 Scope  
The system will be a responsive, CRUD-based web application that:  
- Offers comprehensive IT services like support, managed services, cloud services, development, consulting, and data recovery.  
- Provides clients with inquiry, service request, and feedback options.  
- Includes a secure employee portal with features like attendance, training modules, document submission, salary slip access, and ERP.  
- Supports three user roles: Admin, IT Staff, and Clients.  
- Implements full backend functionality using Node.js, Express.js, MySQL, and Linux deployment, with a frontend in React, HTML, CSS, and JavaScript.

1.3 Technologies Used  
- Frontend: React.js, HTML5, CSS3, JavaScript  
- Backend: Node.js with Express.js  
- Database: MySQL  
- Email System: Nodemailer  
- File Upload: Multer (stored locally)  
- Hosting: Linux Server (optional: Render)  
- Server Management: PM2  
- Authentication: JWT (JSON Web Token)

2. Overall Description  
2.1 Product Perspective  
The system will have two main interfaces:  
- Client Interface: Public website with service descriptions, contact forms, service requests, and feedback submission.  
- Employee Portal: Secure login system with dashboard and role-specific features.

2.2 User Classes and Characteristics  
- Admin: Full access to manage users, services, feedback, complaints, ERP features, training modules, and files.  
- IT Staff: Limited access to assigned tasks, attendance, training, documents, and viewing feedback.  
- Client: Can view services, submit service requests, raise issues, upload documents, and give feedback (no login required).

2.3 Operating Environment  
- Frontend: React, HTML5, CSS3, JavaScript  
- Backend: Node.js with Express.js  
- Database: MySQL  
- Hosting: Linux server  
- Mobile responsive

3. Functional Requirements  
3.1 Client-Side Features  
- View service categories with descriptions  
- Submit service request (support ticket)  
- Submit contact inquiries  
- Submit feedback and rate services  
- Upload documents/images for Admin review  
- Receive auto-reply confirmation email after submission

3.2 Authentication & Authorization  
- Login/Register only for Admin and Staff (Client has no login)  
- Role-based access for Admin and IT Staff  
- Forgot Password feature via Email OTP system for both Admin and Staff

3.3 Admin Features  
- View, add, update, and delete Admins and Staff  
- Assign tasks or service tickets to IT staff  
- View feedback and complaint reports  
- View uploaded client documents and download/delete them  
- Upload documents/salary slips/training files for staff (PDF, images, videos)  
- Search staff by name or ID  
- Track attendance of staff (viewable and searchable)  
- Send email messages to one/specific/all staff from dashboard  
- Forward client complaints to one/specific/all staff  
- View and optionally delete inappropriate feedback

3.4 IT Staff Features  
- View and update assigned service tasks  
- Mark attendance daily and view own records  
- View/download training resources  
- Submit/upload documents  
- Download salary slips  
- Reset password via OTP (email)

3.5 ERP Features  
- Internal employee directory (name, role, contact)  
- Track resource allocations (devices, tools, etc.)  
- Internal announcements/news

4. Non-Functional Requirements  
- Performance: Pages load in <2 seconds for 50+ concurrent users  
- Security: Role-based access, JWT tokens, password hashing  
- Usability: Mobile responsive, intuitive UI with menu-based layout  
- Maintainability: Modular code, MVC structure, RESTful APIs  
- Availability: 99% uptime  
- Compatibility: Modern browsers, responsive on all screen sizes

5. External Interface Requirements  
- UI: Public website (Home, Services, Feedback, Complaint), Employee/Admin dashboards  
- API: Secure REST API (Express.js + MySQL)  
- File Upload: Local server using Multer for storage  
- Email: Send OTP, feedback response, and staff messages via Nodemailer

6. Use Case Examples  
- Client submits a complaint → Admin gets it → Admin forwards it to staff → Staff resolves → Admin notifies client  
- Admin uploads training video → Staff views/downloads it  
- Staff forgets password → Enters email → Receives OTP → Resets password securely

7. Assumptions and Dependencies  
- All users have stable internet access  
- Server supports Node.js and MySQL  
- Email provider (Gmail or SMTP) is accessible  
- Local file storage has sufficient space for uploads

8. Future Enhancements (Optional)  
- Chatbot integration (Tawk.to or Socket.io)  
- Mobile app version  
- Live tracking of tickets